The Double Z Item Buffer for Volume Rendering Non-Regular Grids

Samuel P. Uselton

Report RNR-91-027, September 1991

NAS Systems Division
Applied Research Branch
NASA Ames Research Center
Mail Stop T-045-1
Moffett Field, Ca 94035

Abstract

The Double Z Item Buffer is a technique for improving the speed of volume rendering data sampled over irregular grids. Volume rendering of data sampled over regular grids can be done at interactive speeds and hardware has been designed to increase the size of the grids that can be volume rendered at these speeds. Volume rendering of data sampled on structured (also called array connected), but geometrically irregular, grids and unstructured (also called cell connected) grids is usually done by ray casting (ray tracing limited to initial rays from the eye through the pixels). Ray casting is used because it is a straight-forward method for finding a front-to-back ordering of the cells covering each pixel of the desired image. However, ray casting is numerically intensive and not well supported by the graphics hardware found in workstations. A small modification in graphics microcode and some additional memory (or a small special purpose hardware unit) can be used to generate the front-to-back ordering required for volume rendering data on these inconvenient grids. The same technique can be used to generate a back-to-front ordering, which may be used in compositing operations. The same second Z buffer and front-to-back ordering (or back-to-front ordering) can also be used to implement a more accurate transparency shading for surface modeled objects than is ordinarily available in Z buffer rendering.

S. P. Uselton is employed by Computer Sciences Corporation. This work was performed for NASA under contract NAS-2-12961.

The Double Z Item Buffer for Volume Rendering Non-Regular Grids (Extended Abstract)

Samuel P. Uselton
Computer Sciences Corp.
NASA Ames Research Center

Introduction:

The Double Z Item Buffer is a technique for improving the speed of volume rendering data sampled over irregular grids. Volume rendering of data sampled over regular grids can be done at interactive speeds ([West89], [Levo88]), and hardware has been designed to increase the size of the grids that can be volume rendered at these speeds [Kauf90]. Volume rendering of data sampled on structured (also called array connected), but geometrically irregular, grids and unstructured (also called cell connected) grids is usually done by ray casting (ray tracing limited to initial rays from the eye through the pixels). Ray casting is used because it is a straight-forward method for finding a front-to-back ordering of the cells covering each pixel of the desired image. However, ray casting is numerically intensive and not well supported by the graphics hardware found in workstations. A small modification in graphics microcode and some additional memory (or a small special purpose hardware unit) can be used to generate the front-to-back ordering required for volume rendering data on these inconvenient grids. The same technique can be used to generate a back-to-front ordering, which may be used in compositing operations. The same second Z buffer and front-to-back ordering (or back-to-front ordering) can also be used to implement a more accurate transparency shading for surface modeled objects than is ordinarily available in Z buffer rendering.

Background:

Ray casting, in the context of volume rendering densely sampled data, can be accelerated by exploiting coherency of the grid. For each ray, for any cell after the first one hit, determining which face of the cell contains the exit point of the ray can also be used to determine the next cell entered [Garr90]. Therefore, no global search is required except (1) for the initial cell hit by each ray and (2) for a cell hit re-entering the grid after crossing a hole in the geometry. A strategy for avoiding case (1) was conceived by Hultquist and Uselton, and described by Uselton at the San Diego Workshop on Volume Visualization in December 1990 and in [Usel91]. It is a variation on the Item Buffer idea described by Weghorst, Hooper and Greenberg in [Wegh84] and used in [Sale90]. In this new scheme, each grid cell face is assigned an identification number. In a structured grid this number is formed from the triple of indices of the cell and an indication of which of three possible face orientations. There are six faces per cell in a structured grid, but almost all faces are shared between two cells. A face can be thought of as belonging to the cell with the smaller index values in all three dimensions. This identification number is mapped to RGB color space and cell faces are rendered using a hardware implementation of a traditional Z-buffer visible surface algorithm. Pixels are read from this image and the RGB values mapped back to the cell face identification numbers. As long as all cell faces can be represented uniquely, (with one extra value for the background color) the first cell hit and on which face (and all misses) can be uniquely determined. For unstructured grids, a more arbitrary numbering scheme must be used. A level of indirection is needed to access cell and face information, but otherwise the method is unchanged.

Method:

The Item Buffer technique can be further extended to find successive cells and cell faces covering pixels in depth sorted order. This extension requires a second memory array of the same size as the usual Z buffer. It is used to retain the values of the Z buffer generated in the previous pass through the data. The Double Z Item Buffer method also requires a second Z compare, which can easily be performed in parallel, and an And operation to combine these results. The time required to perform this extra processing, then, is only the additional single gate delay for the And. A psuedocode representation of the algorithm follows.

```
previous_Z = Hither_Clip_Z;
modified = TRUE;
while (modified) do
   Z_buffer = Yon_Clip_Z;
   Image_buffer = background;
   modified = FALSE;
   for (all faces of all cells) do
      for (all pixels covered by face) do
         if ((cell_face_Z(x,y) < Z_buffer(x,y)) and
            (cell_face_Z(x,y) > previous_Z(x,y)) )
            then begin
               Z_buffer(x,y) = cell_face_Z(x,y);
               Image_buffer(x,y) = cell_face_id;
               modified = TRUE;
            end
         end if
      end for
   end for
   /* Image Buffer contains id numbers of cell faces of next layer */

   /* Decode id's and use here */
   previous_Z = Z_buffer; /* actually a pointer value swap */
end while
```

Upon exiting the "for (all faces of all cells)" loop, Image_buffer contains the cell face identification numbers of the faces behind the ones that set the previous_Z values on the previous iteration of the while loop. Repeating this process yields the successive grid cell faces for each pixel. If the then clause is never executed for any pixel in the course of processing the whole grid, then the job is complete. This process yields a front-to-back ordering of grid cell faces for each pixel.

As each layer of cell faces becomes available, the identification numbers must be decoded and data values accessed, and possibly interpolated. Then the shading contribution for this layer can be composited into a separate, final image buffer. Note that the Z values available in the two buffers can be used to determine thickness of the cell along the ray path. This value is important in exponential attenuation volume shading. The result will be the same as if rays had been computed. To maintain accuracy, the hither and yon clipping values should be set as tightly around the grid as possible.

Performance:

The number of repetitions of the main while loop will be one greater than the largest number of grid cell faces covering a pixel. This number is typically on the order of the maximum single dimension of array connected data or the cube root of the number of grid cells in an unstructured grid.

Current polygon rendering speed of graphics workstations is as great as one million polygons per second. The delay to accomplish the And operation in the inner loop should impact performance by no more than 25 or 30 percent, yielding a polygon rate of roughly 700,000 polygons per second. Of course this number does not include the work required to decode and use the cell identification information. Assume that the work in a ray casting implementation like [Garr90] is roughly evenly split between the ray stepping from cell to cell and the data interpolation and shading calculations. We estimate the time spent generating the front-to-back ordering to be less than one tenth of its previous value so the time required to produce similar images would be only slightly more than half that previously required.

Additional uses:

Transparency shading effects can be included in Z buffered polygon rendering by using the same front-to-back ordering of polygons covering each pixel. Alternatively, this use can be optimized by first rendering all opaque polygons and removing them from the list of polygons. Then a back-to-front ordered rendering of transparent polygons can generate the correct shade for each pixel of the image. A back-to-front ordering can be generated by a simple reorganization of the tests used in the code above.

```
if ( (cell_face_Z(x,y) > Z_buffer(x,y)) and
   (cell_face_Z(x,y) < previous_Z(x,y)) )
   then begin
      Z_buffer(x,y) = cell_face_Z(x,y);
      Image_buffer(x,y) = cell_face_id;
      modified = TRUE;
```

The back-to-front ordering is also preferred in some methods of volume rendering. The compositing operation is different, but this arrangement of the tests is unchanged for that application.

Multiple intersecting grids can be displayed using the Double Z Item buffer method with no changes. If one grid has priority over another, as in the fluid dynamics scheme called "I-blanking," slightly more sophistication is required to discard cell faces that are not relevant. The main inner loop, however, is still unchanged. Primitives other than grids, for example radiation beam boundaries, can also be included by simply assigning identification numbers to them.

Bibliography:

[Dreb88] Drebin, Robert A., Loren Carpenter and Pat Hanrahan, "Volume Rendering," Computer Graphics, vol 22, no 4, (August 1988), pp 65-74.

[Garr90] Garrity, Michael P., "Raytracing Irregular Volume Data," Computer Graphics, vol 24, no 5, (November 1990), pp 35-40.

[Kauf90] Kaufman, Arie, Roni Yagel and Reuven Bakalash, "Direct Interaction with a 3D Volumetric Environment," Computer Graphics, vol 24, no 2, (March 1990), pp 33-34.

[Levo88] Levoy, Marc, "Display of Surfaces from Volume Data," IEEE Comp. Graphics and Appl., vol 8, no 3 (May 1988), pp 29-37.

[Neem90] Neeman, Henry, "A Decomposition Algorithm for Visualizing Irregular Grids," Computer Graphics, vol 24, no 5, (November 1990), pp 49-56.

[Sabe88] Sabella, Paolo, "A Rendering Algorithm for Visualizing 3D Scalar Fields," Computer Graphics, vol 22, no 4, (August 1988), pp 51-58.

[Sale90] Salesin, David and Jorge Stolfi, "Rendering CSG Models with a ZZ-Buffer," Computer Graphics, vol 24, no 4, (August 1990), pp 67-76.

[Shir90] Shirley, Peter and Allan Tuchman, "A Polygonal Approximation to Direct Scalar Volume Rendering," Computer Graphics, vol 24, no 5, (November 1990), pp 27-34.

[Upso88] Upson, Craig and Michael Keeler, "VBUFFER: Visible Volume Rendering", Computer Graphics, vol 22, no 4, (August 1988), pp 59-64.

[Usel91] Uselton, Samuel P., "Volume Rendering for Computational Fluid Dynamics: Initial Results," Technical Report RNR-91-026, Applied Research Branch, NAS Systems Division, NASA Ames Research Center, September 1991.

[Wegh84] Weghorst, Hank, Gary Hooper and Donald P. Greenberg, "Improved Computational Models for Ray Tracing," Trans. on Graphics, vol 3, no 1, (Jan. 1984), pp 52-69.

[West89] Westover, Lee, "Interactive Volume Rendering," Proc. Chapel Hill Workshop on Volume Visualization, (May 18-19, 1989), pp 9-16.

[Wilh90] Wilhelms, Jane, Judy Challinger, Naim Alper, Shankar Ramamoorthy, and Arsi Vaziri, "Direct Volume Rendering of Curvilinear Volumes," Computer Graphics, vol 24, no 5, (November 1990), pp 41-48.